МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра

інформатики та програмної інженерії

(повна назва кафедри, циклової комісії)

**КУРСОВА РОБОТА**

з Основ програмування п

(назва дисципліни)

на тему: «Шаховий тренажер»

Студенки 1 курсу, групи ІП-42

Сергієнко Олени Вікторівни

Спеціальності 121 «Інженерія програмного забезпечення»

Керівник

доц. Полупан Ю.В. .

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Національна оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| Члени комісії |  |  |  |
|  | (підпис) |  | (посада, вчене звання, науковий ступінь, прізвище та ініціали) |
|  |  |  |  |
|  | (підпис) |  | (посада, вчене звання, науковий ступінь, прізвище та ініціали) |

Київ – 2025 рік

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

(назва вищого навчального закладу)

Кафедра інформатики та програмної інженерії

Дисципліна Основи програмування. Курсова робота

Напрям «ІПЗ»

Курс 1 Група ІП-42 Семестр 2

ЗАВДАННЯ

на курсову роботу студента

Сергієнко Олени Вікторівни

(прізвище, ім’я, по батькові)

1. Тема роботи Шаховий тренажер

2. Строк здачі студентом закінченої роботи

3. Вихідні дані до роботи додаток А, технічне завдання

4. Зміст пояснювальної записки (перелік питань, які підлягають розробці)

Постановка задачі, теоретичні відомості, опис алгоритмів, опис програмного

забезпечення, тестування програмного забезпечення, інструкція користувача, аналіз

результатів.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

6. Дата видачі завдання

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва етапів курсової роботи | Термін виконання етапів роботи | Підписи керівника, студента |
| 1. | Отримання теми курсової роботи |  |  |
| 2. | Підготовка ТЗ |  |  |
| 3. | Пошук та вивчення літератури з питань курсової роботи |  |  |
| 4. | Розробка сценарію роботи програми |  |  |
| 6. | Узгодження сценарію роботи програми з керівником |  |  |
| 5. | Розробка (вибір) алгоритму розв’язання задач |  |  |
| 6. | Узгодження алгоритму з керівником |  |  |
| 7. | Узгодження з керівником інтерфейсу користувача |  |  |
| 8. | Розробка програмного забезпечення |  |  |
| 9. | Налагодження розрахункової частини програми |  |  |
| 10. | Розробка та налагодження інтерфейсної частини програми |  |  |
| 11. | Узгодження з керівником набору тестів для контрольного прикладу |  |  |
| 12. | Тестування програми |  |  |
| 13. | Підготовка пояснювальної записки |  |  |
| 14. | Здача курсової роботи на перевірку |  |  |
| 15. | Захист курсової роботи |  |  |

Студент

(підпис)

Керівник Полупан Ю.В.

(підпис) (прізвище, ім’я, по батькові)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ р.

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 55 сторінок, 24 рисунки, 28 таблиць, 4 посилання.

Мета роботи: забезпечення розв’язання задач моделювання шахових партій, аналізу стратегій та тактичних маневрів шляхом розробки інтерактивного шахового тренажера.

Вивчено методи розробки шахових програм, проаналізовано основні принципи побудови шахових алгоритмів та їх оптимізації. Розроблено та реалізовано алгоритми методів перевірки валідності ходів, генерації можливих ходів, розпізнавання шаху і мату, аналізу позиції, комп'ютерних опонентів з використанням алгоритмів мінімакс та альфа-бета відсічення для різних рівнів складності, налаштування та збереження шахових позицій, керування шаховим годинником, перетворення пішаків, та рокіровки.

Виконана програмна реалізація алгоритму шахової гри з використанням об'єктно-орієнтованого підходу на мові C#. Реалізовано повноцінний графічний інтерфейс користувача з можливістю перетягування фігур, відображення можливих ходів, шахової нотації, налаштування складності комп'ютерного опонента та збереження/завантаження позицій. Програма підтримує всі стандартні правила шахів, включаючи спеціальні ходи та умови завершення гри.

ШАХОВИЙ ТРЕНАЖЕР.

Зміст

[Вступ 8](#_Toc197907701)

[1 Постановка задачі 10](#_Toc197907702)

[2 Теоретичні відомості 11](#_Toc197907703)

[2.1 Алгоритм Мінімакс (Minimax) 11](#_Toc197907704)

[2.2 Алгоритм Альфа-бета відсічення 13](#_Toc197907705)

[2.3 Паралельний альфа-бета алгоритм 15](#_Toc197907706)

[2.4 Оцінка позиції 17](#_Toc197907707)

[2.5 Вибір алгоритму залежно від складності 19](#_Toc197907708)

[3 Опис алгоритмів 21](#_Toc197907709)

[3.1 Загальний алгоритм 23](#_Toc197907710)

[3.2 Алгоритм перевірки валідності ходу 25](#_Toc197907711)

[3.3 Алгоритм штучного інтелекту для шахів (метод мінімакс з альфа-бета відсіченням) 27](#_Toc197907712)

[3.4 Алгоритм альфа-бета відсічення 27](#_Toc197907713)

[3.5 Алгоритм паралельного альфа-бета відсічення 28](#_Toc197907714)

[3.6 Алгоритм оцінки позиції на дошці 29](#_Toc197907715)

[3.7 Алгоритм перевірки на шах 30](#_Toc197907716)

[3.8 Алгоритм перевірки на шах і мат або пат 30](#_Toc197907717)

[3.9 Алгоритм перетворення пішака 31](#_Toc197907718)

[3.10 Алгоритм завантаження/збереження позиції 32](#_Toc197907719)

[3.11 Алгоритм налаштування позиції (Setup Mode) 33](#_Toc197907720)

[3.12 Алгоритм керування шаховим годинником 34](#_Toc197907721)

[3.13 Алгоритм перевірки валідності шляху фігури 35](#_Toc197907722)

[3.14 Алгоритм обробки перетягування фігури 35](#_Toc197907723)

[3.15 Алгоритм аналізу позиції під атакою 36](#_Toc197907724)

[3.16 Алгоритм шахової нотації 38](#_Toc197907725)

[3.17 Алгоритм обробки подій гри 39](#_Toc197907726)

[3.18 Алгоритм перемикання режимів гри 40](#_Toc197907727)

[3.19 Алгоритм обробки рокіровки 41](#_Toc197907728)

[3.20 Алгоритм визначення можливих ходів для конкретної фігури 42](#_Toc197907729)

[3.21 Алгоритм отримання всіх можливих ходів для гравця 43](#_Toc197907730)

[3.22 Алгоритм зміни кольору гравця в комп'ютерному режимі 43](#_Toc197907731)

[3.23 Алгоритм створення діалогового вікна перетворення пішака 44](#_Toc197907732)

[3.24 Алгоритм ініціалізації дошки 45](#_Toc197907733)

[4 Опис програмного забезпечення 46](#_Toc197907734)

[4.1 Діаграма класів програмного забезпечення 46](#_Toc197907735)

[4.2 Опис методів частин програмного забезпечення 47](#_Toc197907736)

[4.2.1 Стандартні методи 47](#_Toc197907737)

[4.2.2 Користувацькі методи 54](#_Toc197907738)

[5 Тестування програмного забезпечення 79](#_Toc197907739)

[5.1 План тестування 79](#_Toc197907740)

[5.2 Приклади тестування 80](#_Toc197907741)

[6 Інструкція користувача 81](#_Toc197907742)

[6.1 Робота з програмою 81](#_Toc197907743)

[6.2 Формат вхідних та вихідних даних 81](#_Toc197907744)

[6.3 Системні вимоги 81](#_Toc197907745)

[Висновки 83](#_Toc197907746)

[Перелік посилань 84](#_Toc197907747)

[Додаток А Технічне завдання 85](#_Toc197907748)

[Додаток Б Тексти програмного коду 88](#_Toc197907749)

Вступ

Шахи — одна з найдавніших та найінтелектуальніших ігор у світі, що має багатовікову історію та глибоке культурне значення. Перші згадки про шахи датуються VI століттям, коли ця гра зародилася в Індії під назвою «чатуранга». Протягом століть шахи еволюціонували, поширилися по всьому світу та перетворилися не лише на захопливу гру, але й на ефективний інструмент розвитку логіки, стратегічного мислення та концентрації уваги.

У добу стрімкого розвитку цифрових технологій класична шахова гра набула нових форм. Сучасні комп’ютерні шахові програми — це складні системи, засновані на потужних алгоритмах штучного інтелекту, які здатні конкурувати з найсильнішими гросмейстерами. Попри це, значна увага приділяється також створенню навчальних шахових систем, призначених для підтримки гравців різного рівня — від новачків до досвідчених користувачів.

Шаховий тренажер є програмним забезпеченням, яке поєднує функції традиційної шахової програми з навчальними можливостями. Такі системи допомагають вивчати правила гри, розвивати тактичні та стратегічні навички, аналізувати партії й моделювати ігрові ситуації. Однією з ключових переваг є можливість гри проти комп’ютера з різними рівнями складності, що забезпечує індивідуальний темп навчання та поступовий розвиток гравця.

Актуальність розробки шахового тренажера пояснюється низкою чинників. По-перше, шахи мають вагомий освітній потенціал, оскільки сприяють розвитку логічного мислення, пам’яті, зосередженості та здатності до аналізу складних ситуацій. По-друге, на відміну від традиційного навчання, яке передбачає наявність досвідченого тренера, програмний тренажер забезпечує доступ до знань у будь-який зручний час та у будь-якому місці. По-третє, сучасні технології дозволяють адаптувати тренування під індивідуальні потреби користувача. Крім того, такі програми надають широкі аналітичні можливості для глибокого аналізу партій, що значно пришвидшує процес удосконалення гри. Нарешті, цифрові платформи сприяють популяризації шахів серед різних вікових груп, розширюючи аудиторію гри на глобальному рівні.

Метою даної курсової роботи є створення програмного забезпечення шахового тренажера, що дозволяє користувачеві грати в шахи проти комп’ютера з вибором рівня складності, вивчати основи шахової гри, аналізувати партії та моделювати різні ігрові ситуації. Програма матиме зручний графічний інтерфейс і реалізовуватиме всі правила класичних шахів.

Для досягнення цієї мети передбачається виконати такі завдання: вивчити правила гри в шахи та алгоритми їхньої реалізації в програмному забезпеченні; розробити архітектуру програми та реалізувати основну логіку гри, включно з перевіркою на шах, мат, пат та інші ситуації; створити алгоритми штучного інтелекту з різними рівнями складності; реалізувати графічний інтерфейс користувача; а також забезпечити функціонал збереження та завантаження партій, розстановки фігур і ведення історії ходів. Завершальним етапом стане тестування програми в умовах реального використання.

Результати цієї курсової роботи можуть бути корисними як для індивідуального навчання грі в шахи, так і для використання в навчальному процесі освітніх закладів. Окрім того, створений шаховий тренажер може слугувати основою для подальшого розвитку, розширення функціональності та інтеграції нових можливостей.

# Постановка задачі

У межах даної курсової роботи необхідно розробити програмне забезпечення «Шаховий тренажер», що реалізує гру в шахи з можливістю вибору рівня складності суперника. Основною метою є створення інтерактивного застосунку, який дозволить користувачам тренуватись у грі проти комп’ютера або іншого гравця, удосконалюючи свої навички. Передбачається реалізація повного функціоналу класичних шахів, включаючи всі правила пересування фігур, спеціальні шахові прийоми (рокіровку, взяття на проході, перетворення пішака) та умови завершення гри (шах, мат, пат/нічия). Особливу увагу слід приділити реалізації штучного інтелекту, що аналізуватиме поточну позицію та обиратиме оптимальні ходи залежно від обраного рівня складності: простий рівень передбачає випадкові або базові ходи, складніший — застосування алгоритму Мінімакс з альфа-бета відсіканням та обмеженням глибини аналізу.

Програма повинна мати зручний графічний інтерфейс, який забезпечує візуалізацію шахової дошки, можливість переміщення фігур мишею, підсвічування допустимих ходів, ведення історії партії та зручну навігацію в меню. Також має бути реалізована підтримка режиму редагування позиції, збереження і завантаження партій з файлів, можливість скасування ходів, шаховий годинник, відображення захоплених фігур і результату партії. Серед вхідних даних — дії користувача в інтерфейсі (вибір режиму гри, рівня складності, початкової позиції, виконання ходів), а також можливе завантаження позицій з файлу. Вихідними даними є оновлення шахової дошки, повідомлення про стан гри (шах, мат, пат), список ходів у вигляді шахової нотації, індикатори часу та інша візуальна інформація.

# Теоретичні відомості

Шахову дошку як структуру даних можна представити у вигляді двовимірного масиву:

, (2.1)

де:

де A – масив, що представляє шахову дошку, i – індекс рядка (вертикально), j – індекс стовпця (горизонтально).

Кожен елемент A[i,j] може містити одне з наступних значень: 0 – пуста клітинка, додатнє ціле число – біла фігура, від'ємне ціле число – чорна фігура.

Абсолютне значення числа вказує на тип фігури:

1 – пішак, 2 – кінь, 3 – слон, 4 – тура, 5 – ферзь, 6 – король.

## Алгоритм Мінімакс (Minimax)

Алгоритм Мінімакс є фундаментальним методом для знаходження оптимального ходу в шахах. Він ґрунтується на парадигмі "розділяй і володарюй" та використовує рекурсивний підхід для обчислення оцінок ходів.

Нехай:

– поточна позиція на шаховій дошці

– множина всіх можливих ходів у позиції

– функція оцінки позиції з точки зору максимізуючого гравця (білі)

Тоді рекурсивна функція мінімакс визначається як:

де - позиція, що виникає після ходу з позиції , а - глибина пошуку.

Псевдокод алгоритму:

Складність алгоритму

Нехай b – середня кількість можливих ходів з кожної позиції (фактор розгалуження), а d – глибина пошуку.

* Часова складність:
* Просторова складність: для зберігання стеку рекурсії

Метод мінімакс гарантує оптимальний хід при достатній глибині пошуку, але має експоненційну складність, що обмежує його практичне застосування для глибин більше 3-4 при стандартній шаховій позиції.

## Алгоритм Альфа-бета відсічення

Метод альфа-бета відсічення є оптимізацією алгоритму мінімакс. Він дозволяє скоротити кількість вузлів, які необхідно оцінювати, без втрати оптимальності рішення.

Нехай:

- найкраще значення, знайдене для максимізуючого гравця на шляху до поточної позиції

- найкраще значення, знайдене для мінімізуючого гравця на шляху до поточної позиції

Тоді рекурсивна функція альфа-бета визначається як:

з припиненням пошуку, якщо .

Умови відсічення:

Відсічення бета (для максимізуючого гравця): якщо , то решта ходів не може покращити результат.

Відсічення альфа (для мінімізуючого гравця): якщо то решта ходів не може покращити результат.

Псевдокод алгоритму:

Ефективність відсікання залежить від порядку аналізу ходів. Якщо найкращі ходи розглядаються першими, ефективність максимальна.

Теоретично, при оптимальному упорядкуванні ходів, метод альфа-бета відсікання розгляне лише вузлів, де - фактор розгалуження, а - глибина пошуку. Це дозволяє аналізувати дерево пошуку на вдвічі більшу глибину порівняно з чистим мінімаксом.

Ефективність відсікання можна оцінити за допомогою формули:

де – кількість вузлів, що аналізуються алгоритмом мінімакс, а – кількість вузлів, що аналізуються алгоритмом альфа-бета.

У найкращому випадку , де - глибина пошуку.

## Паралельний альфа-бета алгоритм

Паралельний альфа-бета алгоритм використовує переваги багатоядерних процесорів для прискорення пошуку, розподіляючи обчислення різних гілок дерева пошуку між декількома потоками виконання.

Математичне формулювання

де - позиція після застосування ходу до позиції .

Алгоритм паралельно обчислює значення для всіх ходів верхнього рівня і вибирає найкращий.

Псевдокод алгоритму:

Теоретично, коефіцієнт прискорення для процесорів можна оцінити за формулою Амдала:

де - частка обчислень, які можуть бути розпаралелені.

При повному розпаралелюванні верхнього рівня, ідеальне прискорення дорівнює , де - кількість можливих ходів у початковій позиції, а - кількість доступних потоків.

Часова складність: ) у найкращому випадку.

## Оцінка позиції

Усі наведені алгоритми залежать від функції оцінки позиції, яка повертає числове значення, що відображає перевагу одного з гравців.

Математичне формулювання

де:

- функція цінності фігури на позиції

- бонус за шах

Цінність фігур визначається як:

де:

Бонус за шах визначається як:

Псевдокод функції оцінки:

## Вибір алгоритму залежно від складності

Програма вибирає алгоритм для пошуку найкращого ходу залежно від заданої складності.

Математичне формулювання

де:

- поточна позиція

- рівень складності

- множина всіх можливих ходів у позиції

Псевдокод алгоритму вибору:

# Опис алгоритмів

Перелік всіх основних змінних та їхнє призначення наведено в таблиці 3.1.

Таблиця 3.1 – Основні змінні та їхні призначення

| Змінна | Призначення |
| --- | --- |
|  | Внутрішнє представлення 8x8 фігур на дошці |
|  | Відстежує фігури, що вже рухались (для рокіровки) |
|  | Колір гравця, який зараз ходить ("white" або "black") |
|  | Прапорець, що вказує, чи увімкнено режим гри проти комп'ютера |
|  | Прапорець, що вказує, чи грає людина за чорних |
|  | Лічильник для правила 50 ходів |
|  | Генератор випадкових чисел для ШІ |
|  | Функція для відображення діалогу перетворення пішака |
|  | Колекція клітинок шахової дошки для відображення в інтерфейсі |
|  | Історія ходів у шаховій нотації |
|  | Прапорець режиму налаштування позиції |
|  | Обрана фігура для розміщення в режимі налаштування |
|  | Виділена рамка для обраної фігури при розстановці |

Продовження таблиці 3.1

|  |  |
| --- | --- |
| Змінна | Призначення |
|  | Прапорець, що вказує, чи перевернута дошка |
| і | Таймери для шахового годинника |
| і | Час, що залишився для гравців |
|  | Стан активності таймерів |
|  | Поточний рівень складності комп'ютерного опонента |
|  | Об'єкт, що керує логікою гри |
|  | Прапорець активності гри |
|  | Текст результату гри |
|  | Початкова точка при перетягуванні фігури |
|  | Клітинка, яку переміщують під час перетягування |
|  | Прапорець, що вказує, чи завантажена позиція |
|  | Прапорець першого ходу після завантаження позиції |
| і | Кольори клітинок дошки |
| і | Словники для зберігання елементів UI фігур при розстановці |
| і | Панелі для вибору фігур під час налаштування |
|  | Бокова сітка для панелей вибору фігур |
|  | Прапорець режиму гри для двох гравців |

Продовження таблиці 3.1

|  |  |
| --- | --- |
| Змінна | Призначення |
|  | Колір гравця в режимі гри проти комп'ютера |
|  | Прапорець очищення дошки з режиму комп'ютера |
|  | Оцінка ходу |
|  | Прапорці спеціальних ходів (перетворення пішака, рокіровка) |
|  | Інформація про захоплену фігуру |
|  | Обраний тип фігури при перетворенні пішака |
|  | Колір пішака, що перетворюється |

## Загальний алгоритм

1. ПОЧАТОК
2. Ініціалізувати ігрове поле та фігури
   1. Створити шахову дошку розміром 8x8
   2. Встановити початкове розташування фігур на дошці
3. Ініціалізувати параметри гри:
   1. Встановити поточного гравця
   2. Ініціалізувати змінні для відстеження:
      1. Фігур, що рухались
      2. Лічильника для правила 50 ходів
   3. Ініціалізувати режим гри:
      1. ЯКЩО обрано гру проти комп'ютера, ТО встановити , ІНАКШЕ \_
      2. ЯКЩО обрано грати за чорних, ТО встановити , ІНАКШЕ
4. ПОКИ гра активна , ВИКОНУВАТИ:
   1. Відобразити поточний стан дошки на екрані:
      1. ЦИКЛ по всіх рядках дошки ( від 0 до 7):
         1. ЦИКЛ по всіх стовпцях дошки ( від 0 до 7):
            1. Додати клітинку з фігурою (якщо є) до колекції
      2. ЯКЩО , ТО відобразити дошку в перевернутому вигляді
   2. Відобразити інформацію про поточного гравця та статус гри
   3. Обробити хід гравця:
      1. ЯКЩО поточний хід комп'ютера і не хід гравця), ТО:
         1. Виконати метод
      2. ІНАКШЕ очікувати дій від користувача:
         1. ЯКЩО користувач обрав фігуру, ТО:
            1. Перевірити, чи належить фігура поточному гравцю
            2. Показати можливі ходи для обраної фігури
         2. ЯКЩО користувач перемістив фігуру, ТО:
            1. Перевірити валідність ходу за допомогою
            2. ЯКЩО хід валідний, ТО виконати хід:

Перемістити фігуру за допомогою

Перевірити на необхідність перетворення пішака

Оновити історію ходів

Змінити поточного гравця

* 1. Перевірити стан гри:
     1. Перевірити на шах королю поточного гравця за допомогою
     2. Перевірити на шах і мат за допомогою
     3. Перевірити правило 50 ходів
     4. ЯКЩО є умова завершення гри, ТО:
        1. Встановити
        2. Відобразити результат гри у

1. КІНЕЦЬ

## Алгоритм перевірки валідності ходу

1. ПОЧАТОК
2. Перевірити базові умови валідності:
   1. Перевірити, чи знаходяться початкова та кінцева позиції на дошці
   2. Перевірити, чи є фігура на початковій позиції
   3. Перевірити, чи належить фігура поточному гравцю
   4. Перевірити, що на кінцевій позиції немає фігури того ж кольору
3. Визначити тип фігури та перевірити валідність ходу:
   1. ЯКЩО фігура - пішак, ТО:
      1. ЯКЩО хід прямо вперед, ТО:
         1. Перевірити, що клітинка попереду вільна
         2. ЯКЩО це перший хід пішака, ТО дозволити рух на дві клітинки вперед
      2. ЯКЩО хід по діагоналі, ТО перевірити наявність фігури супротивника на кінцевій позиції
   2. ЯКЩО фігура - тура, ТО:
      1. Перевірити, що хід горизонтальний або вертикальний
      2. Перевірити, що шлях не заблокований іншими фігурами
   3. ЯКЩО фігура - кінь, ТО:
      1. Перевірити, що хід відповідає L-подібній формі (2 клітинки в одному напрямку і 1 в перпендикулярному)
   4. ЯКЩО фігура - слон, ТО:
      1. Перевірити, що хід діагональний
      2. Перевірити, що шлях не заблокований іншими фігурами
   5. ЯКЩО фігура - ферзь, ТО:
      1. Перевірити, що хід горизонтальний, вертикальний або діагональний
      2. Перевірити, що шлях не заблокований іншими фігурами
   6. ЯКЩО фігура - король, ТО:
      1. ЯКЩО звичайний хід (на 1 клітинку), ТО перевірити, що хід не більше ніж на 1 клітинку в будь-якому напрямку
      2. ЯКЩО рокіровка (хід на 2 клітинки вбік), ТО:
         1. Перевірити, що ні король, ні відповідна тура ще не рухались
         2. Перевірити, що між королем і турою немає фігур
         3. Перевірити, що король не під шахом і не проходить через поля під шахом
4. Перевірити, чи не залишає хід короля під шахом:
   1. Створити тимчасову копію дошки
   2. Виконати хід на тимчасовій дошці
   3. Перевірити, чи не перебуває король під шахом після ходу
5. Повернути результат перевірки ( - хід валідний, - невалідний)
6. КІНЕЦЬ

## Алгоритм штучного інтелекту для шахів (метод мінімакс з альфа-бета відсіченням)

1. ПОЧАТОК
2. Отримати всі можливі ходи для поточного гравця (комп'ютера)
3. ЯКЩО немає можливих ходів, ТО повернути
4. ЯКЩО обрано рівень складності "Random" (випадковий):
   1. Вибрати випадковий хід з доступних і повернути його
5. ІНАКШЕ виконати пошук кращого ходу з урахуванням глибини (визначається рівнем складності):
   1. ЯКЩО рівень складності високий (Expert або Hard), ТО:
      1. Виконати паралельний альфа-бета алгоритм
   2. ІНАКШЕ ЯКЩО рівень складності середній (Medium або Easy), ТО:
      1. Виконати звичайний альфа-бета алгоритм
   3. ІНАКШЕ:
      1. Виконати простий мінімакс алгоритм
6. КІНЕЦЬ

## Алгоритм альфа-бета відсічення

1. ПОЧАТОК
2. ЯКЩО список можливих ходів порожній АБО глибина пошуку = 0, ТО повернути
3. Встановити найкращий хід = , альфа = мінімально можливе значення, бета = максимально можливе значення
4. ЦИКЛ по всіх можливих ходах:
   1. Створити тимчасову копію дошки
   2. Виконати поточний хід на тимчасовій дошці
   3. ЯКЩО досягнута максимальна глибина пошуку (), ТО:
      1. Оцінити позицію на дошці за допомогою
   4. ІНАКШЕ:
      1. Отримати всі можливі ходи для супротивника
      2. Рекурсивно викликати алгоритм на меншій глибині для супротивника
      3. Отримати оцінку результату рекурсивного виклику
   5. Зберегти оцінку в поточному ході
   6. ЯКЩО гравець - максимізуючий (білі):
      1. ЯКЩО , ТО ,
      2. ЯКЩО , ТО припинити пошук (відсічення бета)
   7. ІНАКШЕ (гравець - мінімізуючий, чорні):
      1. ЯКЩО , ТО ,
      2. ЯКЩО , ТО припинити пошук (відсічення альфа)
5. Повернути найкращий хід
6. КІНЕЦЬ

## Алгоритм паралельного альфа-бета відсічення

1. ПОЧАТОК
2. ЯКЩО список можливих ходів порожній АБО глибина пошуку = 0, ТО повернути
3. Ініціалізувати список оцінених ходів ()
4. ПАРАЛЕЛЬНО ДЛЯ кожного можливого ходу:
   1. Створити копію поточного ходу (для уникнення конфліктів при паралельному виконанні)
   2. Створити тимчасову копію дошки
   3. Виконати поточний хід на тимчасовій дошці
   4. ЯКЩО досягнута максимальна глибина пошуку ТО:
      1. Оцінити позицію на дошці за допомогою
   5. ІНАКШЕ:
      1. Отримати всі можливі ходи для супротивника
      2. Викликати звичайний (не паралельний) алгоритм альфа-бета на меншій глибині
      3. Отримати оцінку результату виклику
   6. Зберегти оцінку в копії поточного ходу
   7. Додати оцінений хід до списку (з синхронізацією)
5. ЯКЩО гравець - максимізуючий (білі), ТО:
   1. Повернути хід з найвищою оцінкою
6. ІНАКШЕ (гравець - мінімізуючий, чорні):
   1. Повернути хід з найнижчою оцінкою
7. КІНЕЦЬ

## Алгоритм оцінки позиції на дошці

1. ПОЧАТОК
2. Ініціалізувати оцінку для білих та чорних
3. ЦИКЛ по всіх рядках дошки ( від 0 до 7):
   1. ЦИКЛ по всіх стовпцях дошки ( від 0 до 7):
      1. Отримати фігуру на позиції
      2. ЯКЩО фігура існує, ТО:
         1. Визначити матеріальну цінність фігури: - Пішак = 1 - Кінь = 3 - Слон = 3 - Тура = 5 - Ферзь = 9 - Король = 100
         2. ЯКЩО колір фігури білий, ТО додати цінність до
         3. ІНАКШЕ додати цінність до
4. ЯКЩО король чорних під шахом, ТО додати 10 до
5. ЯКЩО король білих під шахом, ТО додати 10 до
6. Повернути різницю як загальну оцінку позиції
7. КІНЕЦЬ

## Алгоритм перевірки на шах

1. ПОЧАТОК
2. Знайти позицію короля вказаного кольору за допомогою
3. ЯКЩО король не знайдений, ТО повернути
4. Визначити колір супротивника:
   1. ЯКЩО колір короля "white", ТО колір супротивника = "black"
   2. ІНАКШЕ колір супротивника = "white"
5. Перевірити, чи атакується позиція короля фігурами супротивника:
   1. Перевірити діагональні атаки (ферзь, слон, король, пішак) за допомогою
   2. Перевірити горизонтальні/вертикальні атаки (ферзь, тура, король) за допомогою
   3. Перевірити атаки коня за допомогою
6. ЯКЩО хоча б одна перевірка повернула e, ТО король під шахом (повернути)
7. ІНАКШЕ король не під шахом (повернути )
8. КІНЕЦЬ

## Алгоритм перевірки на шах і мат або пат

1. ПОЧАТОК
2. Отримати всі можливі ходи для вказаного гравця за допомогою
3. ЯКЩО список можливих ходів не порожній, ТО:
   1. Повернути (гра продовжується)
4. ІНАКШЕ (немає доступних ходів):
   1. Перевірити, чи король гравця під шахом за допомогою
   2. ЯКЩО король під шахом, ТО:
      1. Повернути (шах і мат)
   3. ІНАКШЕ:
      1. Повернути (пат)
5. КІНЕЦЬ

## Алгоритм перетворення пішака

1. ПОЧАТОК
2. Перевірити, чи досяг пішак останнього ряду:
   1. ЯКЩО пішак білий І , АБО пішак чорний І , ТО пішак потребує перетворення
3. ЯКЩО пішак потребує перетворення, ТО:
   1. Отримати колір пішака
   2. ЯКЩО режим гри з комп'ютером І хід комп'ютера, ТО:
      1. Автоматично перетворити пішака на ферзя
   3. ІНАКШЕ:
      1. Викликати діалогове вікно для вибору фігури
      2. ЯКЩО користувач скасував операцію, ТО повернути
      3. ІНАКШЕ отримати вибраний тип фігури і виконати перетворення
   4. Оновити відображення дошки
   5. Повернути (перетворення виконано успішно)
4. ІНАКШЕ повернути (пішак не потребує перетворення)
5. КІНЕЦЬ

## Алгоритм завантаження/збереження позиції

1. ПОЧАТОК
2. ДЛЯ операції збереження позиції:
   1. Відкрити діалогове вікно для вибору файлу збереження з розширенням .ches
   2. ЯКЩО файл обрано, ТО:
      1. Записати конфігурацію дошки:
         1. ЦИКЛ по всіх рядках дошки ( від 0 до 7):
            1. Сформувати рядок, що представляє ряд шахової дошки
            2. ЦИКЛ по всіх стовпцях дошки ( від 0 до 7):

ЯКЩО на клітинці є фігура, ТО додати 2-символьний код фігури (перший символ - колір, другий - тип)

ІНАКШЕ додати ".." (пуста клітинка) 2.2.1.1.3.

* + - * 1. Записати рядок у файл
    1. Записати поточного гравця: "CurrentPlayer:[колір]"
    2. Записати історію ходів

1. ДЛЯ операції завантаження позиції:
   1. Відкрити діалогове вікно для вибору файлу з розширенням .ches
   2. ЯКЩО файл обрано, ТО:
      1. Зчитати всі рядки файлу
      2. ЯКЩО у файлі менше 8 рядків, ТО вивести повідомлення про помилку
      3. ІНАКШЕ:
         1. Створити масив Piece?[8, 8] для зберігання стану дошки
         2. ЦИКЛ по перших 8 рядках файлу (рядки дошки):
            1. Розібрати рядок на фігури і додати їх до масиву
         3. Визначити поточного гравця з рядка, що починається з "CurrentPlayer:"
         4. Завантажити історію ходів з файлу
         5. Оновити гру з новим станом дошки і поточним гравцем
2. КІНЕЦЬ

## Алгоритм налаштування позиції (Setup Mode)

1. ПОЧАТОК
2. Встановити
3. Створити панелі для вибору білих та чорних фігур:
   1. Створити кнопки для кожного типу фігур (король, ферзь, тура, слон, кінь, пішак)
   2. Додати обробники подій для вибору фігур
4. Встановити обробники подій для дошки:
   1. КОЛИ користувач обирає фігуру з панелі:
      1. Зберегти обрану фігуру в \_selectedPieceForPlacement
      2. Підсвітити відповідну кнопку
   2. КОЛИ користувач клацає на клітинці дошки:
      1. ЯКЩО обрана фігура (не ):
         1. Перевірити обмеження кількості фігур даного типу
         2. ЯКЩО можна розмістити, ТО поставити обрану фігуру на клітинку
      2. ІНАКШЕ (режим видалення):
         1. Видалити фігуру з клітинки, якщо вона там є
5. КОЛИ користувач завершує налаштування:
   1. Перевірити валідність позиції:
      1. Перевірити наявність білого та чорного короля
      2. ЯКЩО позиція невалідна, ТО вивести повідомлення про помилку
   2. Зберегти позицію в
   3. Оновити
   4. Очистити історію ходів
   5. Встановити
   6. Скинути таймери та розпочати відлік
   7. Встановити , приховати панелі інструментів
6. КІНЕЦЬ

## Алгоритм керування шаховим годинником

1. ПОЧАТОК
2. Ініціалізувати годинники:
   1. Встановити хвилин
   2. Встановити хвилин
   3. Створити таймери для білих і чорних з інтервалом в 1 секунду
3. Запустити таймер для поточного гравця:
   1. Зупинити обидва таймери
   2. ЯКЩО гра активна
      1. ЯКЩО поточний гравець білий, ТО запустити
      2. ІНАКШЕ запустити
4. ДЛЯ обробки тіка таймера (щосекунди):
   1. Зменшити відповідний час або) на 1 секунду
   2. Оновити відображення часу
   3. ЯКЩО час вичерпався (= 0):
      1. Зупинити таймери
      2. Встановити
      3. Показати повідомлення про перемогу протилежного гравця
5. ДЛЯ скидання годинників:
   1. Зупинити обидва таймери
   2. Ініціалізувати годинники заново
   3. Оновити відображення часу
6. КІНЕЦЬ

## Алгоритм перевірки валідності шляху фігури

1. ПОЧАТОК
2. Визначити напрямок руху: 2.1.
   1. , якщо , інакше знак ()
   2. , якщо , інакше знак (
3. Встановити початкову позицію для перевірки:
4. ПОКИ АБО :
   1. ЯКЩО на позиції є фігура, ТО:
      1. Повернути true (шлях заблокований)
   2. Оновити позицію:
5. Повернути (шлях вільний)
6. КІНЕЦЬ

## Алгоритм обробки перетягування фігури

1. ПОЧАТОК
2. КОЛИ користувач натискає на клітинку з фігурою:
   1. Перевірити, чи гра активна
   2. Перевірити, чи це хід гравця
   3. Перевірити, чи фігура належить поточному гравцю
   4. Показати можливі ходи для цієї фігури
   5. ЯКЩО є доступні ходи, ТО:
      1. Зберегти початкову точку перетягування
      2. Зберегти клітинку, яку перетягують
      3. Розпочати операцію перетягування
3. КОЛИ користувач перетягує фігуру над іншою клітинкою:
   1. ЯКЩО клітинка підсвічена (є допустимим ходом), ТО:
      1. Встановити ефект перетягування = Move
   2. ІНАКШЕ:
      1. Встановити ефект перетягування = None
4. КОЛИ користувач відпускає фігуру над клітинкою:
   1. ЯКЩО клітинка підсвічена та не , ТО:
      1. Виконати хід з на цільову клітинку
      2. Очистити підсвічування
      3. Скинути
5. КІНЕЦЬ

## Алгоритм аналізу позиції під атакою

1. ПОЧАТОК
2. Перевірити наявність атаки за трьома різними типами:
   1. Виконати перевірку діагональних атак
      1. Перевірити в чотирьох діагональних напрямках (↖, ↗, ↙, ↘):
         1. Для атаки пішаками:
            1. Визначити напрямок руху пішака залежно від кольору
            2. Перевірити на наявність пішака противника на відповідній діагональній позиції
         2. Для атак дальнобійними фігурами (слон, ферзь):
            1. ЦИКЛ від 1 до 7 клітинок вздовж діагоналі:

ЯКЩО позиція за межами дошки, ТО перейти до наступного напрямку

ЯКЩО на позиції є фігура, ТО:

ЯКЩО це фігура атакуючого кольору І це слон, ферзь або король (на відстані 1), ТО повернути

Перейти до наступного напрямку (фігура блокує подальший шлях)

* 1. Виконати перевірку прямолінійних атак
     1. Перевірити в чотирьох ортогональних напрямках (↑, ↓, ←, →):
        1. ЦИКЛ від 1 до 7 клітинок у кожному напрямку:
           1. ЯКЩО позиція за межами дошки, ТО перейти до наступного напрямку
           2. ЯКЩО на позиції є фігура, ТО:

ЯКЩО це фігура атакуючого кольору І це тура, ферзь або король (на відстані 1), ТО повернути

Перейти до наступного напрямку (фігура блокує подальший шлях)

* 1. Виконати перевірку атак конем
     1. Перевірити всі можливі ходи коня (8 позицій):
        1. Для кожної позиції:
           1. ЯКЩО позиція за межами дошки, ТО пропустити
           2. ЯКЩО на позиції є кінь атакуючого кольору, ТО повернути

1. ЯКЩО будь-яка з перевірок повернула , ТО позиція під атакою
2. ІНАКШЕ позиція не під атакою
3. КІНЕЦЬ

## Алгоритм шахової нотації

1. ПОЧАТОК
2. Отримати інформацію про хід (фігура, початкова і кінцева позиції, захоплена фігура)
3. ЯКЩО це рокіровка (король рухається на 2 клітинки вбік):
   1. ЯКЩО рух вправо, ТО повернути "O-O" (коротка рокіровка)
   2. ІНАКШЕ повернути "O-O-O" (довга рокіровка)
4. ІНАКШЕ сформувати стандартну нотацію:
   1. Визначити символ фігури:
      1. Пішак: "" (порожній символ)
      2. Кінь: "N"
      3. Слон: "B"
      4. Тура: "R"
      5. Ферзь: "Q"
      6. Король: "K"
   2. Визначити символ захоплення: "x" (якщо є захоплена фігура), інакше ""
   3. Отримати коди початкової та кінцевої позицій (наприклад, "e2", "e4")
   4. Перевірити на неоднозначність (коли дві фігури одного типу можуть ходити на ту саму клітинку):
      1. ЦИКЛ по всіх інших фігурах того ж типу і кольору:
         1. ЯКЩО інша фігура також може зробити цей хід, ТО:
            1. Додати уточнення (файл та/або ранг) для розрізнення
   5. Сформувати нотацію: [символ\_фігури][уточнення][символ\_захоплення][кінцева\_позиція]
   6. Перевірити, чи король супротивника під шахом після ходу: 4.6.1. ЯКЩО це мат, ТО додати "#" 4.6.2. ЯКЩО це просто шах, ТО додати "+"
5. Повернути сформовану шахову нотацію
6. КІНЕЦЬ

## Алгоритм обробки подій гри

1. ПОЧАТОК
2. Налаштувати обробники подій з :
   1. BoardUpdated:
      1. Викликати через Dispatcher
   2. :
      1. Виконати через Dispatcher:
         1. Оновити історію ходів ()
         2. Оновити стан дошки ()
   3. :
      1. Виконати через Dispatcher:
         1. Встановити
         2. Зупинити таймери
         3. Показати повідомлення про кінець гри залежно від типу закінчення:
            1. "[Колір] оголосили мат!"
            2. "Пат! Нічия."
            3. : "[Колір] виграли! Король захоплений." 2.3.1.3.4.
            4. : "Нічия за правилом 50 ходів."
         4. Запитати користувача про початок нової гри
         5. ЯКЩО користувач вибрав нову гру, ТО:
            1. Очистити історію ходів
            2. Скинути стан гри
            3. Оновити інтерфейс
   4. PawnPromotion:
      1. Виконати через Dispatcher:
         1. Показати діалогове вікно вибору фігури для перетворення пішака
         2. Встановити вибрану фігуру в аргументах події
         3. ЯКЩО операцію скасовано, ТО встановити
3. КІНЕЦЬ

## Алгоритм перемикання режимів гри

1. ПОЧАТОК
2. ДЛЯ перемикання на режим гри проти комп'ютера:
   1. Запитати користувача про початок нової гри або використання поточної позиції
   2. ЯКЩО користувач вибрав нову гру, ТО скинути гру до початкової позиції
   3. Встановити
   4. Встановити
   5. Передати в
   6. Показати елементи керування складністю комп'ютера
   7. Визначити колір гравця )
   8. Встановити на основі
   9. Скинути таймери
   10. ЯКЩО поточний хід комп'ютера:
       1. Викликати
   11. Оновити інтерфейс
3. ДЛЯ перемикання на режим для двох гравців:
   1. Запитати користувача про підтвердження
   2. ЯКЩО користувач підтвердив, ТО:
      1. Скинути гру до початкової позиції
      2. Очистити історію ходів
      3. Встановити
      4. Встановити
      5. Передати в
      6. Приховати елементи керування складністю комп'ютера
      7. Оновити статусний текст
      8. Скинути таймери
4. КІНЕЦЬ

## Алгоритм обробки рокіровки

1. ПОЧАТОК
2. Перевірити, що переміщувана фігура - король і хід на 2 клітинки вбік
3. ЯКЩО хід відповідає рокіровці:
   1. Перевірити, що король не рухався:
      1. Створити унікальний ідентифікатор короля
      2. Перевірити наявність цього ідентифікатора в
   2. Перевірити, що король на стартовій позиції (стовпець 4, відповідний ряд)
   3. Перевірити, що король не під шахом
   4. ЯКЩО рокіровка вправо (короткий бік):
      1. Перевірити наявність тури на H-файлі
      2. Перевірити, що тура не рухалась
      3. Перевірити відсутність фігур між королем і турою
      4. Перевірити, що король не проходить через клітинки під шахом
   5. ЯКЩО рокіровка вліво (довгий бік):
      1. Перевірити наявність тури на A-файлі
      2. Перевірити, що тура не рухалась
      3. Перевірити відсутність фігур між королем і турою
      4. Перевірити, що король не проходить через клітинки під шахом
   6. ЯКЩО всі умови виконані, ТО дозволити рокіровку
4. Під час виконання рокіровки:
   1. Перемістити короля на 2 клітинки вбік
   2. Визначити, яка це рокіровка (коротка чи довга)
   3. Перемістити відповідну туру на протилежний бік від короля
   4. Позначити туру як переміщену
5. КІНЕЦЬ

## Алгоритм визначення можливих ходів для конкретної фігури

1. ПОЧАТОК
2. Отримати фігуру на вказаній позиції
3. ЯКЩО фігури немає, ТО повернути порожній список
4. Ініціалізувати список
5. Отримати колір фігури
6. ЦИКЛ для всіх можливих позицій на дошці:
   1. ЦИКЛ для від 0 до 7:
      1. ЦИКЛ для від 0 до 7:
         1. ЯКЩО ТО продовжити (пропустити поточну позицію)
         2. ЯКЩО ТО:
            1. Додати до списку
7. Повернути
8. КІНЕЦЬ

## Алгоритм отримання всіх можливих ходів для гравця

1. ПОЧАТОК
2. Ініціалізувати список
3. ЦИКЛ для від 0 до 7:
   1. ЦИКЛ для startCol від 0 до 7:
      1. Отримати фігуру на позиції
      2. ЯКЩО на позиції є фігура І колір фігури = , ТО:
         1. Отримати всі можливі ходи для цієї фігури за допомогою
         2. ЦИКЛ для кожної можливої позиції :
            1. Створити об'єкт Move(startRow, startCol, endRow, endCol)
            2. Додати створений хід до списку allMoves
4. Повернути
5. КІНЕЦЬ

## Алгоритм зміни кольору гравця в комп'ютерному режимі

1. ПОЧАТОК
2. Обробити подію вибору радіокнопки кольору
3. ЯКЩО програма не ініціалізована або не в режимі гри з комп'ютером, ТО вихід
4. Визначити новий колір (White або Black)
5. ЯКЩО колір змінився, ТО:
   1. Показати діалогове вікно з питанням про збереження поточної позиції
   2. Оновити
   3. Оновити
   4. ЯКЩО користувач вибрав не зберігати позицію, ТО:
      1. Ініціалізувати нову гру ()
      2. Оновити інтерфейс дошки
   5. Перевірити, чи зараз хід комп'ютера:
      1. ЯКЩО ( І = "white") АБО (НЕ І = "black"), ТО:
         1. Викликати
   6. Оновити інтерфейс ()
6. КІНЕЦЬ

## Алгоритм створення діалогового вікна перетворення пішака

1. ПОЧАТОК
2. Створити нове вікно
3. Встановити колір пішака () на основі вхідного параметра
4. Встановити заголовок вікна залежно від кольору пішака
5. Створити інтерфейс вибору:
   1. Створити панель для варіантів
   2. Додати опції для кожного типу фігур:
      1. Ферзь (queen)
      2. Тура (rook)
      3. Слон (bishop)
      4. Кінь (knight)
   3. Для кожної опції:
      1. Створити візуальне представлення з іконкою та назвою
      2. Додати обробник події вибору
      3. Встановити ферзя як вибір за замовчуванням
   4. Додати кнопки "OK" та "Відміна"
6. Показати діалогове вікно та чекати вибору користувача
7. ЯКЩО користувач натискає "OK", ТО:
   1. Повернути вибраний тип фігури
8. ЯКЩО користувач натискає "Відміна", ТО:
   1. Встановити
9. КІНЕЦЬ

## Алгоритм ініціалізації дошки

1. ПОЧАТОК
2. Очистити дошку від фігур
3. Розставити фігури білого кольору:
   1. Поставити тури на a1 та h1
   2. Поставити коней на b1 та g1
   3. Поставити слонів на c1 та f1
   4. Поставити ферзя на d1
   5. Поставити короля на e1
   6. Поставити пішаки на другий ряд (від a2 до h2)
4. Розставити фігури чорного кольору:
   1. Поставити тури на a8 та h8
   2. Поставити коней на b8 та g8
   3. Поставити слонів на c8 та f8
   4. Поставити ферзя на d8
   5. Поставити короля на e8
   6. Поставити пішаки на сьомий ряд (від a7 до h7)
5. Очистити список переміщених фігур
6. Викликати подію оновлення дошки
7. КІНЕЦЬ

# Опис програмного забезпечення

## Діаграма класів програмного забезпечення

Діаграма включає наступні ключові класи:

Board - базовий клас, що представляє шахову дошку та містить методи для перевірки ходів, переміщення фігур та оцінки позиції.

Piece - клас, що представляє окрему шахову фігуру з властивостями кольору та типу, а також методами отримання її значення та символьного представлення.

GameLogic - клас, що координує ігровий процес, містить реалізацію алгоритмів штучного інтелекту (мінімакс, альфа-бета), а також керує подіями гри.

BoardCell - клас, що представляє окрему клітинку шахової дошки в інтерфейсі користувача.

MainWindow - головний клас інтерфейсу, що відповідає за відображення гри та обробку подій користувача.

Move - клас, що представляє хід у грі з початковою та кінцевою позиціями.

PawnPromotionDialog - діалогове вікно для вибору фігури при перетворенні пішака.

Діаграма також включає перерахування:

GameEndType - типи завершення гри (мат, пат, тощо)

MoveFlag - типи ходів (звичайний, перетворення пішака, тощо)

ComputerDifficulty - рівні складності комп'ютерного опонента

ChessColor - кольори гравців

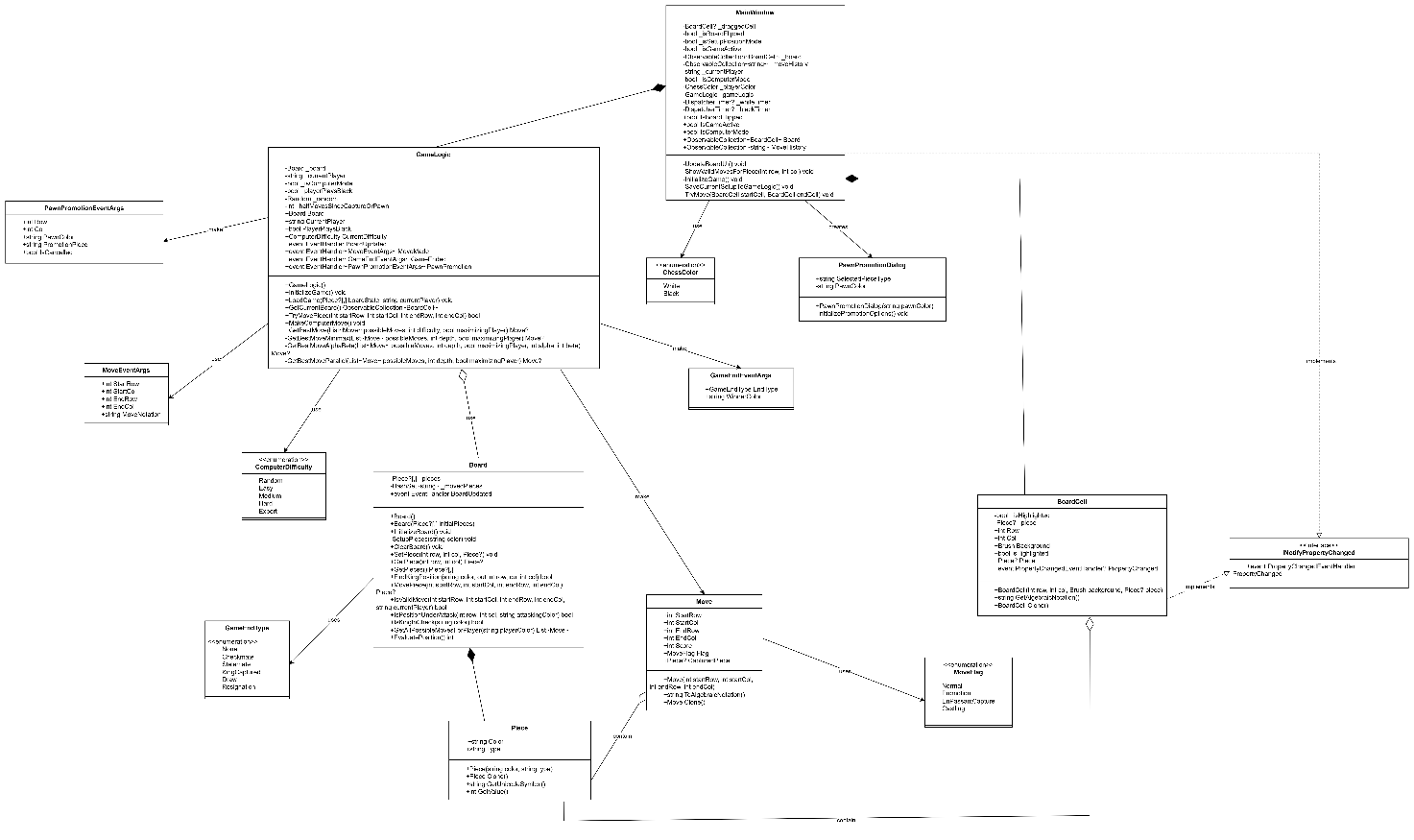


Рисунок 4.1 – Діаграма класів

## Опис методів частин програмного забезпечення

### Стандартні методи

У таблиці 4.1 наведено стандартні методи, які використовуються в проекті шахового тренажера. Ці методи належать до стандартної бібліотеки C# та надають базовий функціонал для взаємодії з користувацьким інтерфейсом, роботи з файловою системою та управління потоками.

Таблиця 4.1 – Стандартні методи

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 1 | MessageBox | Show | Відображення діалогового вікна з повідомленням користувачу | message: текст повідомлення, title: заголовок вікна, button: кнопки діалогу, icon: іконка вікна | MessageBoxResult: результат дії користувача (Yes, No, OK, Cancel) | System.Windows |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 2 | Application | Current.Dispatcher.Invoke | Виконання коду в потоці користувацького інтерфейсу | action: делегат дії, що виконується в потоці UI | void | System.Windows |
| 3 | Application | Current.Dispatcher.InvokeAsync | Асинхронне виконання коду в потоці користувацького інтерфейсу | action: делегат дії, що виконується асинхронно в потоці UI | DispatcherOperation | System.Windows |
| 4 | Task | Run | Асинхронний запуск задачі в окремому потоці | action: делегат дії, що виконується асинхронно | Task: об'єкт задачі | System.Threading.Tasks |
| 5 | File | ReadAllLines | Зчитування всіх рядків текстового файлу | path: шлях до файлу | string[]: масив рядків файлу | System.IO |
| 6 | File | Exists | Перевірка існування файлу | path: шлях до файлу | bool: результат перевірки | System.IO |
| 7 | Random | Next | Генерація випадкового числа | maxValue: максимальне значення (виключно) | int: випадкове число від 0 до maxValue-1 | System |
| 8 | StreamWriter | Write | Запис тексту у файл | text: текст для запису | void | System.IO |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 9 | StreamWriter | WriteLine | Запис рядка тексту у файл з переходом на новий рядок | text: текст для запису | void | System.IO |
| 10 | Parallel | ForEach | Паралельне виконання дії для кожного елемента колекції | source: колекція, body: дія для кожного елемента | void | System.Threading.Tasks |
| 11 | ObservableCollection | Add | Додавання елемента до колекції з сповіщенням | item: елемент для додавання | void | System.Collections.ObjectModel |
| 12 | ObservableCollection | Clear | Очищення колекції з сповіщенням | None | void | System.Collections.ObjectModel |
| 13 | List | Add | Додавання елемента до списку | item: елемент для додавання | void | System.Collections.Generic |
| 14 | List | OrderBy/OrderByDescending | Сортування списку за вказаним критерієм | keySelector: функція вибору ключа сортування | IOrderedEnumerable<T>: відсортована послідовність | System.Linq |
| 15 | HashSet | Add | Додавання елемента до множини | item: елемент для додавання | bool: успішність додавання | System.Collections.Generic |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 16 | HashSet | Contains | Перевірка наявності елемента в множині | item: елемент для перевірки | bool: результат перевірки | System.Collections.Generic |
| 17 | HashSet | Clear | Очищення множини | None | void | System.Collections.Generic |
| 18 | LINQ | FirstOrDefault | Пошук першого елемента, що відповідає умові | predicate: умова пошуку | T: перший елемент або значення за замовчуванням | System.Linq |
| 19 | LINQ | Any | Перевірка наявності елементів, що відповідають умові | predicate: умова перевірки | bool: результат перевірки | System.Linq |
| 20 | LINQ | Where | Фільтрація колекції за умовою | predicate: умова фільтрації | IEnumerable<T>: відфільтрована колекція | System.Linq |
| 21 | DispatcherTimer | Start | Запуск таймера | None | void | System.Windows.Threading |
| 22 | DispatcherTimer | Stop | Зупинка таймера | None | void | System.Windows.Threading |
| 23 | Math | Abs | Отримання абсолютного значення числа | value: числове значення | numeric: абсолютне значення | System |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 24 | Math | Sign | Отримання знаку числа | value: числове значення | int: -1, 0 або 1 | System |
| 25 | EventHandler | Invoke | Виклик обробників події | sender: джерело події, e: аргументи події | void | System |
| 26 | String | IsNullOrEmpty | Перевірка рядка на null або порожнечу | value: рядок для перевірки | bool: результат перевірки | System |
| 27 | String | IsNullOrWhiteSpace | Перевірка рядка на null, порожнечу або лише пробіли | value: рядок для перевірки | bool: результат перевірки | System |
| 28 | String | Substring | Отримання підрядка | startIndex: початковий індекс, length: довжина (опційно) | string: підрядок | System |
| 29 | String | StartsWith | Перевірка початку рядка | value: рядок для перевірки | bool: результат перевірки | System |
| 30 | Type | GetType | Отримання типу об'єкта | None | Type: тип об'єкта | System |
| 31 | Convert | ToString | Конвертація значення в рядок | value: значення для конвертації | string: рядкове представлення | System |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 32 | Grid | SetRow | Встановлення рядка елемента в сітці | element: UI елемент, value: індекс рядка | void | System.Windows.Controls |
| 33 | Grid | SetColumn | Встановлення стовпця елемента в сітці | element: UI елемент, value: індекс стовпця | void | System.Windows.Controls |
| 34 | OpenFileDialog | ShowDialog | Відображення діалогу відкриття файлу | None | bool?: результат вибору | Microsoft.Win32 |
| 35 | SaveFileDialog | ShowDialog | Відображення діалогу збереження файлу | None | bool?: результат вибору | Microsoft.Win32 |
| 36 | Border | SetValue | Встановлення значення властивості | dp: властивість залежності, value: значення | void | System.Windows |
| 37 | Dispatcher | Invoke | Синхронне виконання делегата в потоці диспетчера | callback: делегат для виконання | object: результат виконання | System.Windows.Threading |
| 38 | Window | Close | Закриття вікна | None | void | System.Windows |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 39 | Thread | Sleep | Призупинення виконання потоку на вказаний час | millisecondsTimeout: час в мілісекундах | void | System.Threading |
| 40 | TimeSpan | FromMinutes/FromSeconds | Створення інтервалу часу | value: кількість хвилин/секунд | TimeSpan: інтервал часу | System |
| 41 | TimeSpan | Subtract | Віднімання інтервалу часу | ts: інтервал для віднімання | TimeSpan: результат віднімання | System |
| 42 | DateTime | Now | Отримання поточного часу і дати | None | DateTime: поточний час і дата | System |
| 43 | PropertyChangedEventHandler | Invoke | Виклик обробників події зміни властивості | sender: джерело події, e: аргументи події | void | System.ComponentModel |
| 44 | Visibility | Visible/Collapsed | Значення видимості елементів інтерфейсу | None | Visibility: стан видимості | System.Windows |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 45 | IValueConverter | Convert | Конвертація значення для відображення в UI | value: значення, targetType: цільовий тип, parameter: параметр, culture: культура | object: сконвертоване значення | System.Windows.Data |

### Користувацькі методи

У таблиці 4.2 наведено користувацькі методи, які були розроблені в рамках проекту шахового тренажера. Ці методи забезпечують логіку гри в шахи, обробку ходів, взаємодію з користувацьким інтерфейсом та функціонал штучного інтелекту.

Таблиця 4.2 – Користувацькі методи

| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Board | InitializeBoard | Ініціалізація шахової дошки зі стандартною розстановкою фігур | None | void | Board.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 2 | Board | SetupPieces | Розстановка фігур певного кольору на початкові позиції | color: колір фігур ("white" або "black") | void | Board.cs |
| 3 | Board | ClearBoard | Очищення дошки від усіх фігур | None | void | Board.cs |
| 4 | Board | SetPiece | Встановлення фігури на вказану позицію | row, col: позиція, piece: фігура | void | Board.cs |
| 5 | Board | GetPiece | Отримання фігури з вказаної позиції | row, col: позиція | Piece?: фігура або null | Board.cs |
| 6 | Board | GetPieces | Отримання копії поточного стану дошки | None | Piece?[,]: двовимірний масив фігур | Board.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 7 | Board | FindKingPosition | Знаходження позиції короля вказаного кольору | color: колір короля, row, col: вихідні параметри для позиції | bool: чи знайдено короля | Board.cs |
| 8 | Board | MovePiece | Переміщення фігури на дошці | startRow, startCol: початкова позиція, endRow, endCol: кінцева позиція | Piece?: захоплена фігура (якщо є) | Board.cs |
| 9 | Board | IsValidMove | Перевірка допустимості ходу згідно правил шахів | startRow, startCol: початкова позиція, endRow, endCol: кінцева позиція, currentPlayer: поточний гравець | bool: чи допустимий хід | Board.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 10 | Board | IsValidMoveForPiece | Перевірка допустимості ходу для конкретного типу фігури | startRow, startCol, endRow, endCol, piece: фігура | bool: чи допустимий хід | Board.cs |
| 11 | Board | IsValidPawnMove | Перевірка допустимості ходу пішака | startRow, startCol, endRow, endCol, color: колір пішака | bool: чи допустимий хід | Board.cs |
| 12 | Board | IsValidRookMove | Перевірка допустимості ходу тури | startRow, startCol, endRow, endCol | bool: чи допустимий хід | Board.cs |
| 13 | Board | IsValidKnightMove | Перевірка допустимості ходу коня | absRowDiff, absColDiff: різниця в рядках та стовпцях | bool: чи допустимий хід | Board.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 14 | Board | IsValidBishopMove | Перевірка допустимості ходу слона | startRow, startCol, endRow, endCol, absRowDiff, absColDiff | bool: чи допустимий хід | Board.cs |
| 15 | Board | IsValidQueenMove | Перевірка допустимості ходу ферзя | startRow, startCol, endRow, endCol, absRowDiff, absColDiff | bool: чи допустимий хід | Board.cs |
| 16 | Board | IsValidKingMove | Перевірка допустимості ходу короля | startRow, startCol, endRow, endCol, absRowDiff, absColDiff | bool: чи допустимий хід | Board.cs |
| 17 | Board | IsValidCastling | Перевірка допустимості рокіровки | startRow, startCol, endRow, endCol | bool: чи допустима рокіровка | Board.cs |
| 18 | Board | IsPathBlocked | Перевірка, чи є перешкоди на шляху ходу | startRow, startCol, endRow, endCol | bool: чи є перешкоди | Board.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 19 | Board | MoveWouldLeaveKingInCheck | Перевірка, чи хід залишить короля під шахом | startRow, startCol, endRow, endCol, currentPlayer | bool: чи буде король під шахом | Board.cs |
| 20 | Board | IsPositionUnderAttack | Перевірка, чи клітинка атакована фігурами вказаного кольору | row, col: позиція, attackingColor: колір атакуючих фігур | bool: чи атакована клітинка | Board.cs |
| 21 | Board | IsDiagonallyAttacked | Перевірка діагональних атак на клітинку | row, col, attackingColor | bool: чи атакована діагонально | Board.cs |
| 22 | Board | IsOrthogonallyAttacked | Перевірка вертикальних та горизонтальних атак | row, col, attackingColor | bool: чи атакована ортогонально | Board.cs |
| 23 | Board | IsKnightAttacked | Перевірка атак конем на клітинку | row, col, attackingColor | bool: чи атакована конем | Board.cs |
| 24 | Board | IsKingInCheck | Перевірка, чи перебуває король під шахом | color: колір короля для перевірки | bool: чи перебуває король під шахом | Board.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| №  п/  п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 25 | Board | GetValidMovesForPiece | Отримання всіх допустимих ходів для фігури | row, col: позиція фігури | List<(int, int)>: список позицій для ходів | Board.cs |
| 26 | Board | GetAllPossibleMovesForPlayer | Отримання всіх можливих ходів для гравця | playerColor: колір гравця | List<Move>: список доступних ходів | Board.cs |
| 27 | Board | CheckForGameEnd | Перевірка наявності мату або пату | playerColor: колір гравця для перевірки | GameEndType: тип завершення гри | Board.cs |
| 28 | Board | EvaluatePosition | Оцінка поточної позиції | None | int: числова оцінка (позитивна - перевага білих) | Board.cs |
| 29 | Board | IsValidPosition | Перевірка чи позиція в межах дошки | row, col: позиція для перевірки | bool: чи позиція валідна | Board.cs |
| 30 | Board | ClonePieces | Глибоке копіювання масиву фігур | source, destination: масиви для копіювання | void | Board.cs |
| 31 | Board | OnBoardUpdated | Викликання події оновлення дошки | None | void | Board.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| №  п/  п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 32 | BoardCell | Clone | Створення копії клітинки | None | BoardCell: нова ідентична клітинка | BoardCell.cs |
| 33 | BoardCell | GetAlgebraicNotation | Отримання алгебраїчної нотації для клітинки (e.g., "e4") | None | string: алгебраїчна нотація | BoardCell.cs |
| 34 | BoardCell | ToString | Отримання рядкового представлення клітинки | None | string: рядкове представлення | BoardCell.cs |
| 35 | BoardCell | OnPropertyChanged | Сповіщення про зміну властивості | propertyName: назва властивості | void | BoardCell.cs |
| 36 | BooleanToVisibilityConverter | Convert | Конвертація булевого значення у видимість | value: значення для конвертації, targetType, parameter, culture | object: відповідне значення Visibility | BooleanToVisibiliryConverter.cs |
| 37 | BooleanToVisibilityConverter | ConvertBack | Конвертація видимості у булеве значення | value, targetType, parameter, culture | object: булеве значення | BooleanToVisibiliryConverter.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 38 | GameLogic | InitializeGame | Ініціалізація нової гри | None | void | GameLogic.cs |
| 39 | GameLogic | LoadGame | Завантаження гри з вказаною конфігурацією дошки | boardState: стан дошки, currentPlayer: поточний гравець | void | GameLogic.cs |
| 40 | GameLogic | GetCurrentBoard | Отримання поточного стану дошки як колекції клітинок | None | ObservableCollection<BoardCell>: клітинки дошки | GameLogic.cs |
| 41 | GameLogic | SetComputerMode | Встановлення режиму гри з комп'ютером | isComputerMode: чи включений режим комп'ютера | void | GameLogic.cs |
| 42 | GameLogic | IsPlayerTurn | Перевірка, чи зараз хід гравця | None | bool: чи хід гравця | GameLogic.cs |
| 43 | GameLogic | ClearBoard | Очищення шахової дошки | None | void | GameLogic.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 44 | GameLogic | TryMovePiece | Спроба виконати хід | startRow, startCol: початкова позиція, endRow, endCol: кінцева позиція | bool: чи був хід успішним | GameLogic.cs |
| 45 | GameLogic | IsPawnPromotion | Перевірка чи пішак потребує перетворення | row, col: позиція пішака | bool: чи потрібне перетворення | GameLogic.cs |
| 46 | GameLogic | HandlePawnPromotion | Обробка перетворення пішака | row, col: позиція пішака | bool: чи успішне перетворення | GameLogic.cs |
| 47 | GameLogic | SwitchPlayer | Перемикання ходу на іншого гравця | None | void | GameLogic.cs |
| 48 | GameLogic | GetMoveNotation | Генерування шахової нотації для ходу | piece: фігура, startRow, startCol, endRow, endCol, capturedPiece | string: алгебраїчна нотація ходу | GameLogic.cs |
| 49 | GameLogic | MakeComputerMove | Виконання ходу комп'ютером | None | void | GameLogic.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 50 | GameLogic | GetBestMove | Визначення найкращого ходу для ШІ | possibleMoves: список можливих ходів, difficulty: рівень складності, maximizingPlayer: чи максимізуємо оцінку | Move?: найкращий хід | GameLogic.cs |
| 51 | GameLogic | GetBestMoveMinimax | Пошук найкращого ходу методом мінімакс | possibleMoves, depth, maximizingPlayer | Move?: найкращий хід | GameLogic.cs |
| 52 | GameLogic | GetBestMoveAlphaBeta | Пошук найкращого ходу з використанням альфа-бета відсікання | possibleMoves, depth, maximizingPlayer, alpha, beta | Move?: найкращий хід | GameLogic.cs |
| 53 | GameLogic | GetBestMoveParallel | Паралельний пошук найкращого ходу | possibleMoves, depth, maximizingPlayer | Move?: найкращий хід | GameLogic.cs |
| 54 | GameLogic | OnBoardUpdated | Викликає подію оновлення дошки | None | void | GameLogic.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 55 | GameLogic | OnMoveMade | Викликає подію здійснення ходу | e: аргументи події ходу | void | GameLogic.cs |
| 56 | GameLogic | OnGameEnded | Викликає подію завершення гри | e: аргументи події закінчення гри | void | GameLogic.cs |
| 57 | GameLogic | OnPawnPromotion | Викликає подію перетворення пішака | e: аргументи події перетворення | void | GameLogic.cs |
| 58 | GameLogic | SetPromotionDialogCallback | Встановлення функції зворотного виклику для діалогу перетворення пішака | promotionCallback: функція діалогу | void | GameLogic.cs |
| 59 | MainWindow | InitializeComponent | Ініціалізація компонентів вікна | None | void | MainWindow.xaml.cs |
| 60 | MainWindow | CreateSidePanels | Створення бічних панелей для вибору фігур | None | void | MainWindow.xaml.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 61 | MainWindow | InitializeSetupPanels | Ініціалізація панелей для розстановки фігур | None | void | MainWindow.xaml.cs |
| 62 | MainWindow | OnGameLogicBoardUpdated | Обробник події оновлення дошки | sender, e: аргументи події | void | MainWindow.xaml.cs |
| 63 | MainWindow | OnGameLogicMoveMade | Обробник події здійснення ходу | sender, e: аргументи події ходу | void | MainWindow.xaml.cs |
| 64 | MainWindow | OnGameLogicGameEnded | Обробник події завершення гри | sender, e: аргументи події закінчення гри | void | MainWindow.xaml.cs |
| 65 | MainWindow | OnPawnPromotion | Обробник події перетворення пішака | sender, e: аргументи події | void | MainWindow.xaml.cs |
| 66 | MainWindow | ShowPromotionDialog | Показ діалогу перетворення пішака | pawnColor: колір пішака | string: тип вибраної фігури | MainWindow.xaml.cs |
| 67 | MainWindow | StartNewGame\_Click | Обробник натискання кнопки "Нова гра" | sender, e: аргументи події | void | MainWindow.xaml.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 68 | MainWindow | ClearBoard\_Click | Обробник натискання кнопки "Очистити дошку" | sender, e: аргументи події | void | MainWindow.xaml.cs |
| 69 | MainWindow | SetTwoPlayersMode\_Click | Обробник натискання кнопки режиму двох гравців | sender, e: аргументи події | void | MainWindow.xaml.cs |
| 70 | MainWindow | SetComputerMode\_Click | Обробник натискання кнопки режиму гри з комп'ютером | sender, e: аргументи події | void | MainWindow.xaml.cs |
| 71 | MainWindow | ForceRefreshBoardState | Примусове оновлення стану дошки і UI | None | void | MainWindow.xaml.cs |
| 72 | MainWindow | UpdateStatusAndTimers | Оновлення статусного тексту та таймерів | None | void | MainWindow.xaml.cs |
| 73 | MainWindow | SetupPosition\_Click | Обробник натискання кнопки налаштування позиції | sender, e: аргументи події | void | MainWindow.xaml.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 74 | MainWindow | SavePosition\_Click | Обробник натискання кнопки збереження позиції | sender, e: аргументи події | void | MainWindow.xaml.cs |
| 75 | MainWindow | LoadPosition\_Click | Обробник натискання кнопки завантаження позиції | sender, e: аргументи події | void | MainWindow.xaml.cs |
| 76 | MainWindow | DifficultyComboBox\_SelectionChanged | Обробник зміни рівня складності | sender, e: аргументи події | void | MainWindow.xaml.cs |
| 77 | MainWindow | PlayerColorRadioButton\_Checked | Обробник вибору кольору гравця | sender, e: аргументи події | void | MainWindow.xaml.cs |
| 78 | MainWindow | FlipBoardCheckBox\_Checked | Обробник вибору інверсії дошки | sender, e: аргументи події | void | MainWindow.xaml.cs |
| 79 | MainWindow | BoardCell\_MouseLeftButtonDown | Обробник натискання на клітинку дошки | sender, e: аргументи події | void | MainWindow.xaml.cs |
| 80 | MainWindow | BoardCell\_DragEnter | Обробник входу перетягування у клітинку | sender, e: аргументи події | void | MainWindow.xaml.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 81 | MainWindow | BoardCell\_Drop | Обробник скидання перетягування на клітинку | sender, e: аргументи події | void | MainWindow.xaml.cs |
| 82 | MainWindow | PieceSetup\_Click | Обробник вибору фігури під час налаштування | sender, e: аргументи події | void | MainWindow.xaml.cs |
| 83 | MainWindow | RemovePieceButton\_Click | Обробник кнопки видалення фігури | sender, e: аргументи події | void | MainWindow.xaml.cs |
| 84 | MainWindow | UpdateBoardUI | Оновлення інтерфейсу шахової дошки | None | void | MainWindow.xaml.cs |
| 85 | MainWindow | FlipBoardDisplay | Інверсія відображення дошки | None | void | MainWindow.xaml.cs |
| 86 | MainWindow | UpdateStatusText | Оновлення статусного тексту | None | void | MainWindow.xaml.cs |
| 87 | MainWindow | ClearHighlights | Очищення всіх підсвічувань клітинок дошки | None | void | MainWindow.xaml.cs |
| 88 | MainWindow | ShowValidMovesForPiece | Відображення допустимих ходів для обраної фігури | row, col: позиція фігури | void | MainWindow.xaml.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 89 | MainWindow | PlacePieceInSetupMode | Розміщення фігури в режимі налаштування | clickedCell: клітинка для розміщення | void | MainWindow.xaml.cs |
| 90 | MainWindow | CanPlacePiece | Перевірка чи можна розмістити фігуру (обмеження кількості) | pieceType: тип фігури, pieceColor: колір фігури | bool: чи дозволено розміщення | MainWindow.xaml.cs |
| 91 | MainWindow | GetLocalizedPieceType | Отримання локалізованої назви типу фігури | pieceType: тип фігури | string: локалізована назва | MainWindow.xaml.cs |
| 92 | MainWindow | HighlightSelectedPieceBorder | Підсвічування обраної фігури в панелі налаштувань | selectedBorder: рамка обраної фігури | void | MainWindow.xaml.cs |
| 93 | MainWindow | SetBoardColors | Встановлення кольорів клітинок дошки | lightColor, darkColor: кольори для світлих і темних клітинок | void | MainWindow.xaml.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 94 | MainWindow | UpdateMoveHistory | Оновлення історії ходів | move: хід для додавання в історію | void | MainWindow.xaml.cs |
| 95 | MainWindow | ClearMoveHistory | Очищення історії ходів | None | void | MainWindow.xaml.cs |
| 96 | MainWindow | UpdateMoveHistoryFromLoaded | Оновлення історії ходів з завантажених даних | loadedHistory: завантажена історія | void | MainWindow.xaml.cs |
| 97 | MainWindow | InitializeGame | Ініціалізація гри | None | void | MainWindow.xaml.cs |
| 98 | MainWindow | InitializeBoardUI | Ініціалізація інтерфейсу дошки | None | void | MainWindow.xaml.cs |
| 99 | MainWindow | CreatePieceSetupButton | Створення кнопки для розстановки фігур | pieceColor, pieceType: параметри фігури, panel: панель розміщення | void | MainWindow.xaml.cs |
| 100 | MainWindow | SaveCurrentSetupToGameLogic | Збереження поточної розстановки в GameLogic | None | void | MainWindow.xaml.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 101 | MainWindow | TryMove | Спроба виконати хід в інтерфейсі | startCell, endCell: початкова та кінцева клітинки | void | MainWindow.xaml.cs |
| 102 | MainWindow | LoadGameFromFile | Завантаження гри з файлу | lines: рядки файлу з даними гри | void | MainWindow.xaml.cs |
| 103 | MainWindow | ParseBoardRow | Розбір рядка даних дошки з файлу | line: рядок з файлу, row: індекс рядка, boardState: стан дошки для заповнення | void | MainWindow.xaml.cs |
| 104 | MainWindow | DecodePieceOldFormat | Декодування формату фігури зі старого формату файлу | code: код фігури | Piece?: фігура або null для порожньої клітинки | MainWindow.xaml.cs |
| 105 | MainWindow | DecodePieceNewFormat | Декодування формату фігури з нового формату файлу | code: код фігури | Piece?: фігура або null для порожньої клітинки | MainWindow.xaml.cs |
| 106 | MainWindow | InitializeTimers | Ініціалізація шахових годинників | None | void | MainWindow.xaml.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 107 | MainWindow | CreateTimer | Створення таймера з вказаним обробником | tickHandler: обробник тіків таймера | DispatcherTimer: створений таймер | MainWindow.xaml.cs |
| 108 | MainWindow | StartTimers | Запуск шахових годинників | None | void | MainWindow.xaml.cs |
| 109 | MainWindow | StopTimers | Зупинка шахових годинників | None | void | MainWindow.xaml.cs |
| 110 | MainWindow | ResetTimers | Скидання шахових годинників | None | void | MainWindow.xaml.cs |
| 111 | MainWindow | WhiteTimerTick | Обробник тіку таймера білих | sender, e: аргументи події | void | MainWindow.xaml.cs |
| 112 | MainWindow | BlackTimerTick | Обробник тіку таймера чорних | sender, e: аргументи події | void | MainWindow.xaml.cs |
| 113 | MainWindow | UpdateTimeLeft | Оновлення залишеного часу гравця | timeLeft: посилання на час, textBlock: текстовий блок для відображення, gameOverMessage: повідомлення про кінець часу | void | MainWindow.xaml.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 114 | MainWindow | UpdateTimersDisplay | Оновлення відображення таймерів | None | void | MainWindow.xaml.cs |
| 115 | MainWindow | SetProperty | Встановлення властивості з повідомленням про зміну | field: поле для зміни, newValue: нове значення, propertyName: назва властивості | void | MainWindow.xaml.cs |
| 116 | MainWindow | OnPropertyChanged | Викликання події зміни властивості | propertyName: назва зміненої властивості | void | MainWindow.xaml.cs |
| 117 | MainWindow | ShowConfirmation | Показ діалогу підтвердження | message: повідомлення | bool: результат підтвердження | MainWindow.xaml.cs |
| 118 | MainWindow | GetPieceTypeName | Отримання назви типу фігури | pieceType: тип фігури | string: назва типу | MainWindow.xaml.cs |
| 119 | Move | ToAlgebraicNotation | Отримання алгебраїчної нотації для ходу | None | string: алгебраїчна нотація ходу | Move.cs |
| 120 | Move | ToString | Отримання рядкового представлення ходу | None | string: рядкове представлення | Move.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 121 | Move | GetSquareNotation | Перетворення позиції на алгебраїчну нотацію | col, row: координати клітинки | string: алгебраїчна нотація | Move.cs |
| 122 | Move | Clone | Створення глибокої копії ходу | None | Move: копія ходу | Move.cs |
| 123 | Piece | Clone | Створення копії фігури | None | Piece: нова ідентична фігура | Piece.cs |
| 124 | Piece | ToString | Отримання рядкового представлення фігури | None | string: рядкове представлення | Piece.cs |
| 125 | Piece | GetUnicodeSymbol | Отримання Unicode символу для фігури | None | string: Unicode символ | Piece.cs |
| 126 | Piece | GetValue | Отримання матеріальної цінності фігури | None | int: числова цінність | Piece.cs |
| 127 | Piece | GetLocalizedTypeName | Отримання локалізованої назви типу фігури | None | string: локалізована назва | Piece.cs |
| 128 | Piece | GetLocalizedColorName | Отримання локалізованої назви кольору | None | string: локалізована назва | Piece.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 129 | Piece | GetLocalizedName | Отримання повної локалізованої назви фігури | None | string: локалізована назва | Piece.cs |
| 130 | PawnPromotionDialog | InitializeComponent | Ініціалізація компонентів діалогу | None | void | PawnPromotionDialog.xaml.cs |
| 131 | PawnPromotionDialog | InitializePromotionOptions | Ініціалізація опцій для перетворення пішака | None | void | PawnPromotionDialog.xaml.cs |
| 132 | PawnPromotionDialog | AddPieceOption | Додавання опції вибору фігури у діалозі | panel: панель для розміщення, pieceType: тип фігури, fontFamily: шрифт | void | PawnPromotionDialog.xaml.cs |
| 133 | PawnPromotionDialog | GetPieceLocalizedName | Отримання локалізованої назви типу фігури | pieceType: тип фігури | string: локалізована назва | PawnPromotionDialog.xaml.cs |
| 134 | PawnPromotionDialog | Option\_Checked | Обробник вибору опції перетворення | sender, e: аргументи події | void | PawnPromotionDialog.xaml.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 135 | PawnPromotionDialog | OkButton\_Click | Обробник натискання кнопки "OK" | sender, e: аргументи події | void | PawnPromotionDialog.xaml.cs |
| 136 | PawnPromotionDialog | CancelButton\_Click | Обробник натискання кнопки "Відміна" | sender, e: аргументи події | void | PawnPromotionDialog.xaml.cs |
| 137 | PieceConverter | Convert | Конвертація даних фігури у візуальне представлення | values: масив значень (колір та тип), targetType, parameter, culture | object: візуальне представлення | PieceConverter.cs |
| 138 | PieceConverter | GetUnicodeSymbol | Отримання Unicode символу для фігури | color: колір фігури, type: тип фігури | string: символ Unicode | PieceConverter.cs |
| 139 | PieceConverter | ConvertBack | Зворотна конвертація (не реалізована) | value, targetTypes, parameter, culture | object[]: масив сконвертованих значень | PieceConverter.cs |
| 140 | MoveEventArgs | Конструктор | Створення нового об'єкту аргументів події ходу | startRow, startCol, endRow, endCol, moveNotation | None (конструктор) | GameLogic.cs |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 141 | GameEndEventArgs | Конструктор | Створення нового об'єкту аргументів події завершення гри | endType: тип завершення, winnerColor: колір переможця | None (конструктор) | GameLogic.cs |
| 142 | PawnPromotionEventArgs | Конструктор | Створення нового об'єкту аргументів події перетворення пішака | row, col: позиція, pawnColor: колір пішака | None (конструктор) | GameLogic.cs |

# Тестування програмного забезпечення

## План тестування

Передмова (має бути обов’язково).

1. Тестування правильності введених значень.
   1. Тестування при введенні некоректних символів.
   2. Тестування при введенні замалих та завеликих значень.
2. Тестування коректної роботи при введені систем, що не мають коренів.
   1. Тестування роботи програми при нульовому значенні визначника.
   2. Тестування роботи методу 1 на несиметричній матриці.
   3. Тестування роботи методу 1 на недодатньо визначеній матриці.
3. Тестування коректності роботи методів 1,2,3.
   1. Перевірка коректності роботи методу 1.
   2. Перевірка коректності роботи методу 2.
   3. Перевірка коректності роботи методу 3.
4. Тестування коректності роботи методів 1, 2, 3 з дробовими коефіцієнтами.
   1. Перевірка правильності результатів.
   2. …
5. Тестування побудови графіків.
6. …

## Приклади тестування

Передмова + посилання на таблиці тестів.

Таблиця 5.1 – Приклад роботи програми при введенні некоректних символів

|  |  |
| --- | --- |
| Мета тесту | Перевірити можливість введення некоректних даних |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | 2 3 b 6 S 4 6 f f y 9 17 |
| Схема проведення тесту | Поелементне заповнення матриці коефіцієнтів |
| Очікуваний результат | Повідомлення про помилку  формату даних |
| Стан програми після проведення випробувань | Видано помилку «Введіть дійсне число» |

# Інструкція користувача

## Робота з програмою

Після запуску виконавчого файлу з розширенням \*.exe, відкривається головне вікно програми (Рисунок 6.1).

Рисунок 6.1 – Головне вікно програми

Далі за допомогою лічильника з назвою «Розмір системи» шляхом натиску на стрілки або введенням числа з клавіатури необхідно виставити розмір системи, що буде оброблятися програмою (рисунок 6.2):

Рисунок 6.2 – Вибір необхідного розміру системи

## Формат вхідних та вихідних даних

Користувачем на вхід програми подається СЛАР у матричному вигляді, тобто задається за допомогою матриці системи та стовпця вільних членів, числа яких дійсні з точністю не більше, ніж 3 знака після коми (якщо точність більша, то програма автоматично округлить їх за математичними правилами до 3-х знаків після коми).

Результатом виконання програми є розв’язок зданої СЛАР, який видається у вигляді таблиці кожне число якої записане з точністю до 3-х знаків після коми або повідомлення, що дана система не має розв’язків або не сходиться для обраного методу.

## Системні вимоги

Системні вимоги до програмного забезпечення наведені в таблиці 6.1.

Таблиця 6.1 – Системні вимоги програмного забезпечення

|  |  |  |
| --- | --- | --- |
|  | Мінімальні | Рекомендовані |
| Операційна система | Windows® XP/Windows Vista/Windows 7/ Windows 8/Windows 10 (з останніми обновленнями) | Windows 7/ Windows 8/Windows 10  (з останніми обновленнями) |
| Процесор | Intel® Pentium® ІІІ  1.0 GHz або  AMD Athlon™ 1.0 GHz | Intel® Pentium® D або AMD Athlon™ 64 X2 |
| Оперативна пам'ять | 256 MB RAM (для Windows® XP) / 1 GB RAM (для Windows Vista/Windows 7/  Windows 8/Windows 10) | 2 GB RAM |
| Відеоадаптер | Intel GMA 950 з відеопам'яттю об'ємом не менше 64 МБ (або сумісний аналог) | |
| Дисплей | 800х600 | 1024х768 або краще |

Продовження таблиці 6.1

|  |  |  |
| --- | --- | --- |
|  | Мінімальні | Рекомендовані |
| Прилади введення | Клавіатура, комп’ютерна миша | |
| Додаткове програмне забезпечення | Microsoft .Net Framework 4.5.2 або вище | |

Висновки

Коротко описати, що було виконано в кожному розділі. Можливо описати шляхи покращення розробленого програмного забезпечення.

Перелік посилань

1. Russel S., Norvig P. "Artificial Intelligence: A Modern Approach". -Pearson, 2020. - 1136 p.
2. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. "Introduction to Algorithms". - MIT Press, 2022. - 1312 p.
3. Silver D., Hubert T., Schrittwieser J., et al. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play". - Science, 2018. - Vol. 362, Issue 6419. - pp. 1140-1144.
4. Shannon C.E. "Programming a Computer for Playing Chess". - Philosophical Magazine, 1950. - Vol. 41, Issue 314. - pp. 256-275.
5. Knuth D.E., Moore R.W. "An analysis of alpha-beta pruning". - Artificial Intelligence, 1975. - Vol. 6, Issue 4. - pp. 293-326.
6. Стокфіш. Офіційний сайт шахової програми. [Електронний ресурс]. - Режим доступу: https://stockfishchess.org/
7. Šablatura J. "How does Stockfish work: An evaluation of the neural network". - Journal of Artificial Intelligence Research, 2023. - Vol. 74. - pp. 1101-1132.
8. Троєльсен Е. "Мова програмування C# та платформа .NET". - К.: Діалектика, 2020. - 1328 c.
9. Інтелектуальні програми для аналізу шахових партій. - [Електронний ресурс]. - Режим доступу: https://www.chess.com/analysis
10. Bramer M. "Computer Game Playing: Theory and Practice". - Springer, 2018. - 278 p.

Додаток А Технічне завдання

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра

інформатики та програмної інженерії

Затвердив

Керівник \_\_Головченко М.М.\_\_\_\_

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_202\_ р.

Виконавець:

Студент \_\_\_Сергієнко О.В.*\_\_\_\_*

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_202\_ р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання курсової роботи

на тему: «Шаховий тренажер»

з дисципліни:

«Основи програмування. Курсова робота»

Київ 2025

* 1. *Мета*: Метою курсової роботи є забезпечення розв’язання задач моделювання шахових партій, аналізу стратегій та тактичних маневрів шляхом розробки інтерактивного шахового тренажера. Програма повинна надавати користувачам інструменти для гнучкого налаштування позицій на дошці, включаючи встановлення, переміщення та видалення фігур відповідно до правил гри.
  2. *Дата початку роботи*: «26» лютого 2025 р.
  3. *Дата закінчення роботи*: «25» травня 2025 р.
  4. *Вимоги до програмного забезпечення*.

1. Функціональні вимоги:

* Можливість моделювання шахових позицій;
* Можливість гри у режимі проти комп’ютера;
* Можливість гри для двох гравців на одному екрані;
* Можливість перевірки правильності ходів;
* Можливість візуалізації шахової дошки та фігур;
* Можливість підсвічування можливих ходів;
* Можливість виявлення шаху, мату та пату;
* Можливість очищення шахової дошки;
* Можливість налаштування складності гри проти комп’ютера;
* Можливість перегляду історії ходів;
* Можливість зміни кольору шахів;
* Можливість моделювання допустимої ігрової ситуації з можливістю подальшого розіграшу;
* Можливість тайм-контролю.

1. Нефункціональні вимоги:

* Процесор: Intel(R) Core(TM) i5-865U;
* Об’єм ОЗП: 8 ГБ;
* Операційна система: Windows 11;
* Мова програмування: С#;
* Все програмне забезпечення та супроводжуюча технічна документація повинні задовольняти наступним ДЕСТам:

ДСТУ 3008 - 2015 - Розробка технічної документації.

* 1. *Стадії та етапи розробки програмного забезпечення*:

1. Розробка алгоритмічної складової програмного забезпечення (до\_\_.\_\_.202\_ р.)
2. Об’єктно-орієнтований аналіз предметної області завдання (до\_\_.\_\_.202\_ р.)
3. Об’єктно-орієнтоване проєктування програмного забезпечення (до \_\_.\_\_.202\_р.)
4. Розробка програмного забезпечення (до \_\_.\_\_.202\_р.)
5. Тестування розробленого програмного забезпечення (до \_\_.\_\_.202\_р.)
6. Демонстрація та захист програмного забезпечення (до \_\_.\_\_.202\_ р.)
   1. *Порядок контролю та приймання*. Поточні результати роботи над КР регулярно демонструються викладачу. Своєчасність виконання основних етапів графіку підготовки роботи впливає на оцінку за КР відповідно до критеріїв оцінювання.

Додаток Б Тексти програмного коду

*студента групи ІП-ХХ І курсу*

*Гуменського В.Л.*

(Обсяг програми (документа), арк., Кб)

*182 арк, 124 Кб*

(Вид носія даних)

*GitHub репозиторій*

(Найменування програми (документа))

*Тексти програмного коду програмного забезпечення вирішення задачі обернення матриць*

Тут вставляється назва файлу та його вміст